2011 American Control Conference
on O'Farrell Street, San Francisco, CA, USA
June 29 - July 01, 2011

Fast Distributed Consensus with Chebyshev Polynomials

Eduardo Montijano, Juan I. Montijano, and Carlos Sagues

Abstract— **Global observation of the environment is a key component in sensor networks and multi-robot systems. Distributed consensus algorithms make all the nodes in the network to achieve a common perception by local interactions between direct neighbors. The convergence rate of these algorithms depends on the network connectivity, which is related to the second largest eigenvalue of the weighted adjacency matrix of the communication graph. When the connectivity is small, a large number of communication rounds is required to achieve the consensus. In this paper we present a new distributed consensus algorithm which uses the properties of Chebyshev polynomials to significantly increase the convergence rate. The algorithm is expressed in the form of a linear iteration and, at each step, the nodes only require to transmit their current state to their neighbors. The difference with respect to previous approaches is that our algorithm is based on a second order difference equation. We provide the analytical expression of the convergence rate and we study in which conditions it is faster than computing the powers of the weighted matrix. This improvement reduces the number of messages between nodes, saving both power and time to the networked system. We evaluate our algorithm in a simulated environment showing the benefits of our approach.**

Index Terms - Distributed consensus, Convergence rate, Chebyshev polynomials.

I. Introduction

Sensor networks and multi-robot systems are a current topic of interest due to the amount of possibilities they can offer to humankind. Global perception of the environment is one of the key components in these systems. A common estimation of the world can be computed by the network in a distributed fashion using consensus algorithms [1]–[5]. The previous algorithms have in common that they asymptotically reach agreement on the observations of the nodes considering only local interactions. The number of communication rounds required to reach the agreement depends on the network connectivity, characterized by the second largest eigenvalue of the weighted adjacency matrix associated to the network. As the number of nodes in the network grows, communications between different pairs become more difficult due to distance and power constraints, which implies less connectivity and many iterations before achieving consensus.

This work was supported by the project DPI2009-08126 and grant AP2007-03282 Ministerio de Educacion y Ciencia.

E. Montijano and C. Sagues are with Departamento de Informática e Ingeniería de Sistemas - Instituto de Investigación en Ingeniería de Aragón (I3A), Universidad de Zaragoza, Spain. emonti@unizar.es, csagues@unizar.es

J.I. Montijano is with Departamento de Matemática Aplicada - Instituto Universitario de Matemáticas y Aplicaciones (IUMA), Universidad de Zaragoza, Spain. monti@unizar.es

In order to reduce the number of iterations, we present a new discrete time consensus algorithm based on the distributed evaluation of Chebyshev polynomials [6] of the first kind. The algorithm is expressed in the form of a linear iteration using the weighted adjacency matrix and at each step the nodes only require to transmit their current state to their neighbors. The difference with respect to previous approaches is that our algorithm is based on a second order difference equation, using not only the current local status but also the previous one. In this way the convergence rate is increased by a significant factor. This improvement reduces the number of messages between nodes, saving time and energy.

The topic of fast convergence is not new to the research community. The appropriate selection of the weights of the adjacency matrix is discussed in [7]. That work provides the optimal weights for the matrix, as well as good approximations that do not require any global knowledge about the network topology. Our algorithm makes use of the same matrices but in a second order difference equation requiring less iterations to obtain the consensus value with the same error tolerance. A multi-hop protocol is presented in [8]. By sending additional information in the messages their solution improves convergence rates. Compared to this approach, ours has the advantage that it only requires to transmit one data to the neighbors at each time instant.

Second order recurrences to speed up convergence are used in [9], [10]. Both approaches use fixed gains in the algorithms to mix the data whereas with our approach the coefficients are updated at each iteration using the Chebyshev polynomials recurrence. The use of polynomials to speed up convergence has been treated in [11]–[13]. The approach in [11] uses a polynomial of fixed degree with coefficients computed assuming the network is known. A distributed algorithm to compute the minimal polynomial of the adjacency matrix is used in [12] and [13]. Once the polynomial is known, the network can achieve the consensus in a number of communication rounds upper bounded by the size of the network. In small networks this solution works pretty well and has the advantage of getting the exact solution, except the rounding errors. However, as we show in the paper, when the number of nodes grows, the rounding errors in the evaluation of the polynomial can make the algorithm unstable. In contrast with these approaches, our algorithm presents asymptotic convergence but it is stable for any number of nodes.

Finally, there are other approaches that present solutions to achieve consensus in finite time. Continuous time solutions

978-1-4577-0079-8/11/$26.00 ©2011 AACC

can be found in [14], [15]. These approaches are also non linear, which makes them usually hard to be implemented in discrete time systems. In these solutions the number of iterations is also affected by the use of numerical integrators because they depend on the number of steps taken by the method. The approach in [16] proposes a link scheduling that reaches the desired result in a finite number of steps. In wireless networks, links are constrained by the distance between the nodes, which implies that not all the pairs of nodes will be able to communicate and there might be situations in which this method cannot be used.

In the paper we provide the analytical expression of the convergence rate of our algorithm and we study when it is faster than algorithms based on the powers of the adjacency matrix. Experiments with synthetic data show the benefits of using our algorithm compared to other approaches. Along the paper we will consider a fixed communication topology, leaving the analysis of time varying networks for future work.

The structure of the paper is the following: In section II we review the consensus algorithms based on the weighted adjacency matrix and the minimal polynomial. In section III the new consensus algorithm using Chebyshev polynomials is described. Section IV analyzes the behavior of the algorithm in a simulated setup. Finally in section V the conclusions of the work are presented.

II. Distributed Consensus Algorithms

In the paper we consider a sensor network of N agents labeled by $i \in \mathcal{V} = \{1, \ldots, N\}$. Communications between the agents are defined according to an undirected graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ represents the edge set. In this way, agents i and j can communicate if and only if $(i, j) \in \mathcal{E}$. The neighbors of one agent $i \in \mathcal{V}$ are the subset of agents that can directly communicate with it; i.e., $\mathcal{N}_i = \{j \in \mathcal{V} \mid (i, j) \in \mathcal{E}\}$. We assume that the communication graph is fixed and connected, that is, there exists a path of one or more links between any two agents in the network. For simplicity in the notation we assume that each agent has an initial value $x_i \in \mathbb{R}$, however, all the equations and algorithms along the paper are valid for data of any arbitrary dimension. Defining $\bar{x}$ as the average of all the initial conditions, we say that average consensus is achieved by the network when $x_i = x_j = \bar{x}, \forall i, j \in \mathcal{V}$. In practice, average consensus will be achieved when $|x_i - \bar{x}| < \texttt{tol}$ for all i, for a prefixed error tolerance `tol`.

The discrete time distributed consensus algorithm based on the weighted adjacency matrix associated to the communication graph [1] is

$$x_i(n+1) = a_{ii} x_i(n) + \sum_{j \in \mathcal{N}_i} a_{ij} x_j(n), \tag{1}$$

with $x_i(0) = x_i$. The algorithm can also be expressed in vectorial form as

$$\mathbf{x}(n+1) = \mathbf{A}\mathbf{x}(n), \tag{2}$$

where $\mathbf{x}(n) = (x_1(n), \ldots, x_N(n))^T$ and $\mathbf{A} = [a_{ij}] \in \mathbb{R}^{N \times N}$, is the weighted matrix, which satisfies:

***Assumption** 2.1 (Doubly Stochastic Weights):* $\mathbf{A}$ is symmetric, row stochastic and compatible with the underlying graph, $\mathcal{G}$, i.e., it is such that $a_{ii} > 0,\ a_{ij} = 0$ if $(i, j) \notin \mathcal{E}$, $a_{ij} > 0$ only if $(i, j) \in \mathcal{E}$ and $\mathbf{A}\mathbf{1} = \mathbf{1}$.

Since the communication graph is connected, if $\mathbf{A}$ fulfills assumption 2.1, then it has one eigenvalue $\lambda_1 = 1$ with associated right eigenvector $\mathbf{1}$ and algebraic multiplicity equal to one. The rest of the eigenvalues, sorted in decreasing order, satisfy $1 > \lambda_2 \geq \ldots \geq \lambda_N > -1$. Without loss of generality, let us suppose that all the eigenvalues are simple. Any initial conditions $\mathbf{x}(0)$ can be expressed as a sum of eigenvectors of $\mathbf{A}$,

$$\mathbf{x}(0) = \mathbf{v}_1 + \ldots + \mathbf{v}_N,$$

where $\mathbf{v}_i$ is an eigenvector associated to the eigenvalue λ_i. It is clear that

$$\mathbf{x}(n) = \mathbf{A}^n \mathbf{x}(0) = \mathbf{v}_1 + \lambda_2^n \mathbf{v}_2 + \ldots + \lambda_N^n \mathbf{v}_N,$$

and since $|\lambda_i| < 1,\ i \neq 1$, $\lim_{n \to \infty} \mathbf{x}(n) = \mathbf{v}_1 = \gamma \mathbf{1}$, with $\gamma = 1/N \mathbf{1}^T \mathbf{x}(0)$, and average consensus is reached by all the agents in the network. The convergence speed of (2) depends on $\max(|\lambda_2|, |\lambda_N|)$. When the size of the network is large or the number of links is small this value is usually close to one, which means that the algorithm requires many iterations before obtaining a good approximation of the final solution.

To overcome this limitation, the algorithm proposed in [12] computes a linear combination of the first N initial values of $\mathbf{x}(n),\ n = 0, 1, \ldots, N-1$, that provides the desired final solution. The combination is obtained from the minimal polynomial of $\mathbf{A}$. The minimal polynomial of $\mathbf{A}$, $p_{\mathbf{A}}(x)$, with roots equal to the eigenvalues of $\mathbf{A}$, is the polynomial with minimum degree such that $p_{\mathbf{A}}(\mathbf{A}) = 0$. The polynomial can be decomposed into $p_{\mathbf{A}}(x) = (x-1) q_{\mathbf{A}}(x)$ because 1 is a simple eigenvalue of $\mathbf{A}$, and $q_{\mathbf{A}}(1) = 1$. The roots of $q_{\mathbf{A}}(x)$ are the rest of the eigenvalues of $\mathbf{A}$. If the polynomial is evaluated in $\mathbf{A}$ we have

$$\begin{aligned} q_{\mathbf{A}}(\mathbf{A})\mathbf{x}(0) &= \alpha_0 \mathbf{x}(0) + \alpha_1 \mathbf{A}\mathbf{x}(0) + \ldots + \alpha_{N-1} \mathbf{A}^{N-1} \mathbf{x}(0) \\ &= \alpha_0 \mathbf{x}(0) + \alpha_1 \mathbf{x}(1) + \ldots + \alpha_{N-1} \mathbf{x}(N-1) \\ &= q_{\mathbf{A}}(1)\mathbf{v}_1 + q_{\mathbf{A}}(\lambda_2)\mathbf{v}_2 + \ldots + q_{\mathbf{A}}(\lambda_N)\mathbf{v}_N = \mathbf{v}_1, \end{aligned} \tag{3}$$

with α_i the coefficients of the polynomial. Therefore, knowing the coefficients α_i of $q_{\mathbf{A}}(x)$, with $N-1$ iterations the final consensus is obtained.

In absence of rounding errors, the finite-time consensus algorithm provides the exact solution in at most $N-1$ iterations. However, when the number of agents is large or the connectivity is small the coefficients of the polynomial $q_{\mathbf{A}}(x)$ can be very large, which makes the evaluation of $q_{\mathbf{A}}(x)$ numerically instable (see e.g. [17]) and the rounding errors can invalidate the solution. Examples of this situation are provided in section IV.

III. Consensus Using Chebyshev Polynomials

Following the idea of [12] we would like to find polynomials, $p_n(x)$, that can be evaluated in a distributed way satisfying that for all $n > 0$, $p_n(1) = 1$ and $\max_{x\in(-1,1)} |p_n(x)| < 1$. The proposed polynomials must also have a stable evaluation independently on the number of nodes in the network or the connectivity. Let us note that eq. (2) can be written as $\mathbf{x}(n) = w_n(\mathbf{A})\mathbf{x}(0)$ with $w_n(x) = x^n$, which evidently satisfies that $\max_{x\in(-1,1)} |w_n(x)| < 1$ and $w_n(1) = 1, \forall n > 0$. In addition, the proposed polynomial must satisfy $p_n(\lambda_i) \to 0$ when $n \to \infty$ in a faster way than $\max(|\lambda_2|, |\lambda_N|)^n$ which implies that $\mathbf{x}(n) = p_n(\mathbf{A})\mathbf{x}(0) \simeq \mathbf{v}_1$ for smaller n than the algorithm using eq. (2).

Our consensus algorithm is based on Chebyshev polynomials of first kind. These polynomials are defined with the recurrence:

$$\begin{aligned} &T_0(x) = 1 \\ &T_1(x) = x \\ &T_n(x) = 2xT_{n-1}(x) - T_{n-2}(x), \quad n = 2, 3, \dots. \end{aligned} \tag{4}$$

Chebyshev Polynomials have the following properties:

- $T_n(x) = \cos(n \arccos x) \leq 1$ for $x \in [-1, 1]$
- $\max_{x\in[-1,1]} |T_n(x)| = 1$, $T_n(1) = 1$, $T_n(-1) = (-1)^n$ and $|T_n(x)| > 1$ for all $|x| > 1$

***Lemma** 3.1:* The direct expression of $T_n(x)$ is characterized by

$$T_n(x) = \frac{1 + \tau^{2n}}{2\tau^n}, \ \tau = \tau(x) = x - \sqrt{x^2 - 1}. \tag{5}$$

Looking at eq. (5), it is clear that if $x > 1$, then $\tau(x) < 1$ and $T_n(x)$ goes to infinity as n grows. On the other hand, if $|x| < 1$, then $\tau(x)$ is a complex number with $|\tau(x)| = 1$. In this case $|T_n(x)| \leq 1$, $\forall n$. Moreover, for a given n, by the first property of $T_n(x)$, in the interval $x \in [-1, 1]$ there are $n + 1$ points for which $|T_n(x)| = 1$. This implies that $T_n(x)$ is not smaller than 1 for all $x \in (-1, 1)$ and we can not ensure convergence to the desired result if we use this polynomial for the consensus process.

We overcome this limitation by doing a linear transformation that brings some interval $[\lambda_m, \lambda_M]$ to $[-1, 1]$. In this way, given two real coefficients λ_m, λ_M, satisfying $1 > \lambda_M > \lambda_m > -1$, we define the polynomial

$$p_n(x) = \frac{T_n(cx - d)}{T_n(c - d)}, \tag{6}$$

with

$$c = \frac{2}{\lambda_M - \lambda_m}, \quad d = \frac{\lambda_M + \lambda_m}{\lambda_M - \lambda_m}, \tag{7}$$

which has the following properties:

- if $x \in [\lambda_m, \lambda_M]$, then $cx - d \in [-1, 1]$
- $p_n(1) = 1$ and $p_n(\lambda_M + \lambda_m - 1) = (-1)^n$
- $\max_{x\in(\lambda_M+\lambda_m-1,1)} |p_n(x)| < 1$
- $|p_n(x)| > 1$ for all $x \notin [\lambda_M + \lambda_m - 1, 1]$
- The polynomials $p_n(x)$ satisfy the recurrence

$$\begin{aligned} p_{n+1}(x) = 2\frac{T_n(c - d)}{T_{n+1}(c - d)}(cx - d)p_n(x) - \\ - \frac{T_{n-1}(c - d)}{T_{n+1}(c - d)}p_{n-1}(x) \end{aligned} \tag{8}$$

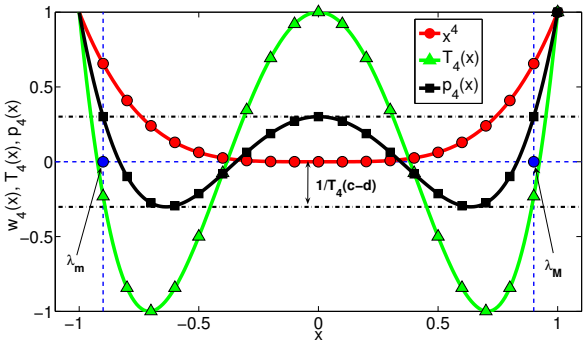


Fig. 1. Plot of the polynomials $w_n(x)$, $T_n(x)$ and $p_n(x)$. In the figure $n = 4, \lambda_m = -0.95$ and $\lambda_M = 0.95$.

Figure 1 shows a plot of $w_n(x)$, $T_n(x)$ and $p_n(x)$, for $n = 4$, in the interval $[-1, 1]$. Note that $T_n(x)$ cannot be used in the consensus process because at some points it does not reduce the error. On the other hand, $p_n(x)$ satisfies the conditions required to achieve consensus. It is also interesting to note that $p_n(x)$ has closer values to zero than $w_n(x)$ in points close to -1 and 1, which means that the error associated to eigenvalues in that regions will be reduced faster.

The recurrence relation for $p_n(x)$ allows us to define the new consensus rule $\mathbf{x}(n) = p_n(\mathbf{A})\mathbf{x}(0)$ by

$$\begin{aligned} \mathbf{x}(1) &= \frac{1}{T_1(c - d)}(c\mathbf{A}\mathbf{x}(0) - d\mathbf{x}(0)), \\ \mathbf{x}(n + 1) &= 2\frac{T_n(c - d)}{T_{n+1}(c - d)}(c\mathbf{A}\mathbf{x}(n) - d\mathbf{x}(n)) - \\ &\quad - \frac{T_{n-1}(c - d)}{T_{n+1}(c - d)}\mathbf{x}(n - 1). \end{aligned} \tag{9}$$

This consensus rule allows a stable computation of successive iterations in a distributed way only by transmitting the current state to the neighbors, as in (2). The only additional information required in the algorithm are λ_m and λ_M. Before stating the main properties of the new algorithm we introduce an auxiliary result to proof the convergence:

***Lemma** 3.2:* Given $x_1 > 1$, for any x_2 such that $|x_2| < x_1$ it holds that

$$\lim_{n\to\infty} \frac{T_n(x_2)}{T_n(x_1)} = 0. \tag{10}$$

***Theorem** 3.1 (Convergence of the algorithm):* For any $\mathbf{A}$ fulfilling Assumption 2.1 and parameters λ_m and λ_M such that $1 > \lambda_M > \lambda_m > -1$ and $\lambda_N > \lambda_m + \lambda_M - 1$, the recurrence in eq. (9) converges to the average of the initial conditions, $\lim_{n\to\infty} \mathbf{x}(n) = \mathbf{v}_1$. Besides,

- The convergence rate is given by

$$\|\mathbf{x}(n) - \mathbf{v}_1\|_2 \leq \max_{\lambda_i \neq 1} \frac{|T_n(c\lambda_i - d)|}{T_n(c - d)} \|\mathbf{x}(0) - \mathbf{v}_1\|_2. \tag{11}$$

- If all the eigenvalues of $\mathbf{A}$, apart from λ_1, are in the interval $[\lambda_m, \lambda_M]$, then

$$\|\mathbf{x}(n) - \mathbf{v}_1\|_2 \leq \frac{1}{T_n(c - d)} \|\mathbf{x}(0) - \mathbf{v}_1\|_2. \tag{12}$$

Proof. Let $\mathbf{Q} = \mathbf{A} - \frac{1}{N}\mathbf{1}\mathbf{1}^T$, whose eigenvalues are $\lambda_1 = 0$, with $\mathbf{v}_1$ its corresponding eigenvector, and $\lambda_2, \ldots, \lambda_N$ with the same eigenvectors as $\mathbf{A}$. Since $\mathbf{v}_1 = \frac{1}{N}\mathbf{1}^T\mathbf{x}(0)\mathbf{1}$, then $\frac{1}{N}\mathbf{1}\mathbf{1}^T(\mathbf{x}(0) - \mathbf{v}_1) = 0$. Taking this into account it is easy to see that

$$\mathbf{A}^k(\mathbf{x}(0) - \mathbf{v}_1) = \mathbf{Q}^k(\mathbf{x}(0) - \mathbf{v}_1), \quad \forall k \in \mathbb{N}, \tag{13}$$

and therefore $p_n(\mathbf{A})(\mathbf{x}(0) - \mathbf{v}_1) = p_n(\mathbf{Q})(\mathbf{x}(0) - \mathbf{v}1)$.

Also $\mathbf{A}\mathbf{v}_1 = \mathbf{v}_1$ and $p_n(1) = 1$, then $p_n(\mathbf{A})\mathbf{v}_1 = \mathbf{v}_1$ and

$$\begin{aligned}&\|\mathbf{x}(n) - \mathbf{v}_1\|_2 = \|p_n(\mathbf{A})(\mathbf{x}(0) - \mathbf{v}_1)\|_2 = \\ &\|p_n(\mathbf{Q})(\mathbf{x}(0) - \mathbf{v}_1)\|_2 \leq \|p_n(\mathbf{Q})\|_2\|\mathbf{x}(0) - \mathbf{v}_1\|_2.\end{aligned} \tag{14}$$

In addition, $\mathbf{Q}$ is symmetric, and so is $p_n(\mathbf{Q})$, which implies that its spectral norm coincides with the spectral radius,

$$\|p_n(\mathbf{Q})\|_2 = \rho(p_n(\mathbf{Q})) = \max_{i\neq 1}|p_n(\lambda_i)| = \max_{i\neq 1}\frac{|T_n(c\lambda_i - d)|}{T_n(c-d)}. \tag{15}$$

For any $x \in (\lambda_M + \lambda_m - 1, 1)$ we have that $|cx - d| < c - d$, then for all the eigenvalues of $\mathbf{A}$ but λ_1, $|c\lambda_i - d| < c - d$. Finally, noting that $c - d$ is strictly larger than 1, by Lemma 3.2 $p_n(\lambda_i) \to 0$ for all $i \neq 1$, which proves the convergence of the algorithm.

Now, if $\lambda_i \in [\lambda_m, \lambda_M]$ for all $i \neq 1$, $|c\lambda_i - d| \leq 1$ and

$$\max_{i\neq 1}|p_n(\lambda_i)| \leq 1/T_n(c-d). \tag{16}$$

■

Note that the conditions in Theorem 3.1 are easy to fulfill without the necessity of knowing the eigenvalues of the matrix $\mathbf{A}$. A safe choice of parameters is $-\lambda_m = \lambda_M,\ 0 < \lambda_M < 1$, which always satisfies the required conditions. It is also convenient to choose $\lambda_M \simeq 1$ to ensure that all the eigenvalues are contained in $[\lambda_m, \lambda_M]$. Further discussion on the selection of the parameters is given in Section IV. We show now when the new algorithm achieves the consensus faster than approaches based on the powers of the adjacency matrix.

***Theorem** 3.2 (Faster convergence than $\mathbf{A}^n$):* For any matrix $\mathbf{A}$ satisfying Assumption 2.1, let $\lambda = \max(|\lambda_2|, |\lambda_N|)$ be the convergence rate in (2). For any

$$0 < \lambda_M < \frac{2\lambda}{\lambda^2 + 1}, \text{ and } \lambda_m = -\lambda_M, \tag{17}$$

$p_n(\lambda)$ goes to zero faster than λ^n when n goes to infinity. Therefore the algorithm in eq. (9) converges faster to the average than the one in eq. (2).

Proof. By eq. (7) we have that

$$c - d = 1/\lambda_M,\ c\lambda - d = \lambda/\lambda_M \tag{18}$$

To prove the faster convergence of our algorithm we show that the quotient between λ^n and $p_n(\lambda)$ goes to infinity with n. That is

$$\lim_{n\to\infty}\frac{\lambda^n}{p_n(\lambda)} = \lim_{n\to\infty}\frac{\lambda^n T_n(c-d)}{T_n(c\lambda - d)} = \infty. \tag{19}$$

Using Lemma 3.1 we substitute the value of $T_n(c-d)$ and $T_n(c\lambda - d)$

$$\frac{\lambda^n}{p_n(\lambda)} = \lambda^n\frac{\tau(c\lambda - d)^n}{\tau(c-d)^n}\ \frac{1 + \tau(c-d)^{2n}}{1 + \tau(c\lambda - d)^{2n}}, \tag{20}$$

where τ is the function in eq. (5). The second term of (20) goes to one as n goes to infinity. This means that, in order to fulfill (19) it must hold that

$$\lim_{n\to\infty}\left(\frac{\lambda\tau(c\lambda - d)}{\tau(c-d)}\right)^n = \infty \Leftrightarrow \left|\frac{\lambda\tau(c\lambda - d)}{\tau(c-d)}\right| > 1. \tag{21}$$

Replacing τ for its value and doing some calculations using the radical conjugates we obtain

$$\frac{\lambda\tau(c\lambda - d)}{\tau(c-d)} = \lambda\frac{1 + \sqrt{1 - \lambda_M^2}}{\lambda + \sqrt{\lambda^2 - \lambda_M^2}}. \tag{22}$$

Using (22) we obtain that (21) is equivalent to

$$\begin{aligned}\lambda_M^2(1 - \lambda^2) > 0, \text{ if } \lambda_M \leq \lambda \\ \lambda_M(-1 - \lambda^2) + 2\lambda > 0, \text{ otherwise}\end{aligned} \tag{23}$$

which by (17) is always true, and therefore, the proof is complete. ■

The last issue we analyze is the selection of λ_m and λ_M to maximize the convergence speed in the situation in which all the eigenvalues are included in the interval $[\lambda_m, \lambda_M]$.

***Proposition** 3.1 (Choice of the parameters):* The best values for λ_m and λ_M such that $[\lambda_N, \lambda_2] \subseteq [\lambda_m, \lambda_M]$ coincide with the minimum and maximum eigenvalues of $\mathbf{A}$, excluding λ_1, that is

$$\lambda_M = \lambda_2, \text{ and } \lambda_m = \lambda_N \tag{24}$$

Proof. The convergence speed is determined by how fast $T_n(c-d)^{-1}$ goes to zero. Since $T_n(x)$ is increasing for $x > 1$, the larger $c - d$ is, the faster the convergence speed. Now

$$c - d = \frac{2 - \lambda_M - \lambda_m}{\lambda_M - \lambda_m}, \tag{25}$$

which is decreasing for λ_M. Therefore, the optimal value of λ_M is λ_2, because we still require that $[\lambda_N, \lambda_2] \subseteq [\lambda_m, \lambda_M]$.

Once λ_M is fixed we can take the derivative of (25) with respect to λ_m

$$\frac{\partial(c-d)}{\partial\lambda_m} = \frac{2 - 2\lambda_M}{(\lambda_M - \lambda_m)^2}, \tag{26}$$

which is always positive, and then $c - d$ is an increasing function for λ_m. Therefore the optimum value of λ_m will be the maximum one, and this yields to $\lambda_m = \lambda_N$. ■

To conclude this section, Algorithm 1 shows a possible implementation of the algorithm. The implementation is quite similar to that of (1), so that current distributed applications using that algorithm can upgrade to our approach without much effort, obtaining the benefits of a faster convergence.

IV. Simulations

We have also analyzed our algorithm in a simulated environment. Several Monte Carlo experiments have been designed to study the convergence of the method and the influence of the parameters λ_m and λ_M in the algorithm.

Algorithm 1 Consensus Algorithm - Agent i

Require: $x_i(0)$, MaxIt $\in \mathbb{N}$, λ_m, λ_M, a_{ij}, $j \in \mathcal{N}_i$

- $\mathbf{A} = [a_{ij}]$ satisfies assumption 2.1
- $1 > \lambda_M > \lambda_m > -1$, equal for all $i \in \mathcal{V}$
- $\lambda_N > \lambda_m + \lambda_M - 1$

Ensure: $x_i^{(2)} \rightarrow \bar{x} = 1/N\mathbf{1}^T\mathbf{x}(0)$ when MaxIt $\rightarrow \infty$

– *Initialization*

$c = 2/(\lambda_M - \lambda_m)$; $\quad d = (\lambda_M + \lambda_m)/(\lambda_M - \lambda_m)$;

$a^{(0)} = 1$; $\quad x_i^{(0)} = x_i(0)$;

$a^{(1)} = c - d$;

– *First Communication Round*

$$x_i^{(1)} = \frac{1}{a^{(1)}}(c \sum_{j \in \mathcal{N}_i} a_{ij} x_j^{(0)} + (c\ a_{ii} - d)x_i^{(0)});$$

repeat

$a^{(2)} = 2(c-d)a^{(1)} - a^{(0)}$;

– *Communication Between Neighbors*

$$x_i^{(2)} = 2\frac{a^{(1)}}{a^{(2)}}(c \sum_{j \in \mathcal{N}_i} a_{ij} x_j^{(1)} + (c\ a_{ii} - d)x_i^{(1)}) - \frac{a^{(0)}}{a^{(2)}} x_i^{(0)};$$

– *Update Parameters*

$a^{(0)} = a^{(1)}$; $\quad x_i^{(0)} = x_i^{(1)}$;

$a^{(1)} = a^{(2)}$; $\quad x_i^{(1)} = x_i^{(2)}$;

until MaxIt

A. Numerical instability of the minimal polynomial

Before the evaluation of our algorithm, we show the numerical instability of the methods based on the minimal polynomial of $\mathbf{A}$ [12] for a sufficiently large network, like the one in Figure 2 (a), which has 35 nodes. The matrix $\mathbf{A}$ has been computed using the "local degree weights" [7]. The minimal polynomial has been computed directly from $\mathbf{A}$ to show that, even with an exact polynomial, the numerical errors can make the algorithm fail. We have generated random initial conditions in the interval $(0, 10)$ for each node and we have used (3) to evaluate the polynomial. In Fig. 2 (b) we show the values of $x_i(N)$ for the different nodes. The numerical errors in the evaluation of the minimal polynomial disturb the final results. On the other hand, in Fig. 2 (c) we have plotted the evolution of $p_n(\mathbf{A})\mathbf{x}(0)$. In this case the evaluation is stable and converges to the average of the initial measurements (black dashed line). Moreover, for a tolerance error of 10^{-3} the algorithm requires less than N iterations to converge.

B. Evaluation of the convergence speed of our algorithm

Let us now evaluate how our algorithm behaves compared to the powers of the weighted adjacency matrix. In this experiment we have analyzed 100 random networks of 50 nodes. For each network the nodes have been randomly positioned in a square of 200×200 meters. Two nodes communicate if they are at a distance lower than 20 meters. The networks are also forced to be connected so that the algorithms converge. After that 100 different random initial values have been generated in the interval $(0, 1)^N$, giving a total of 10000 trials to test the different algorithms.

For each communication network we have computed 3 different weighted adjacency matrices. The first one, $\mathbf{A}_{ld}$, uses the "local degree weights", the second one, $\mathbf{A}_{bc}$, uses the "best constant factor" and the third one, $\mathbf{A}_{os}$, computes an approximation of the "optimal symmetric weights". For more information about these matrices we refer the reader to [7]. We have analyzed the convergence speed of the powers of these matrices and the speed of our algorithm using the same matrices, $p_n(\mathbf{A}_{ld}), p_n(\mathbf{A}_{bc})$ and $p_n(\mathbf{A}_{os})$. In each case we have assigned the parameters $\lambda_M = \lambda_2$ and $\lambda_m = \lambda_N$ to the algorithm. We have measured the average number of iterations required to obtain an error, $e = \|\mathbf{x}(n) - \mathbf{v}_1\|_2$, smaller than a given tolerance.

Table I shows the results of the experiment. For any tolerance error our algorithm reaches the desired precision in far less iterations than (2). Using the same matrix, our algorithm obtains the same results reducing by one order of magnitude the number of iterations (e.g., for $\mathbf{A}_{ld}$ and a tolerance error of $e < 10^{-3}$, (2) requires 823.8 iterations and $p_n(\mathbf{A}_{ld})$ only requires 57.5). Another interesting detail is that our algorithm converges faster using the "local degree weights", $\mathbf{A}_{ld}$, than the other two matrices, even though the second largest eigenvalue of the other two matrices is smaller. This behavior is caused because the interval that contains all the eigenvalues of $\mathbf{A}_{ld}$ is smaller than the intervals that contain the eigenvalues of $\mathbf{A}_{bc}$ and $\mathbf{A}_{os}$ (an example can be found in [7]). As a consequence, $c - d$ is larger and the algorithm converges faster. This is indeed very convenient because the "local degree weights" can be easily computed without global information, whereas the other two require the knowledge of the topology.

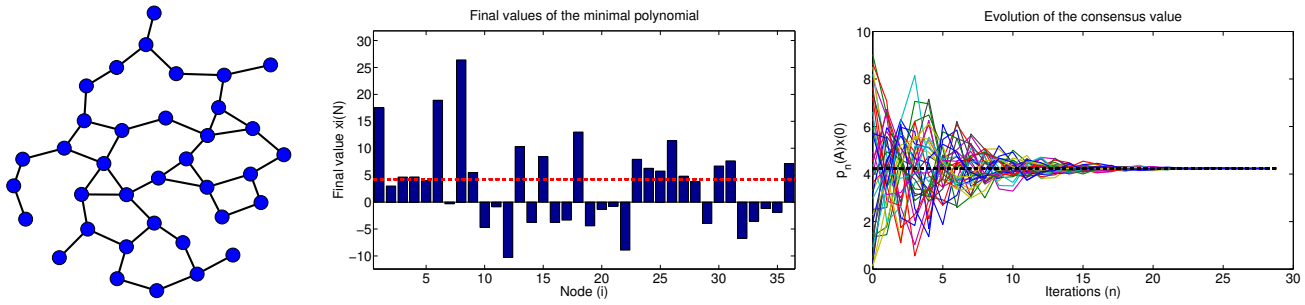
TABLE I
NUMBER OF ITERATIONS FOR DIFFERENT ALGORITHMS

Method\Tolerance	10^{-2}	10^{-3}	10^{-4}	10^{-5}
$\mathbf{A}_{ld}^n$	467.3	823.8	1191.0	1555.5
$\mathbf{A}_{bc}^n$	408.8	679.1	950.7	1222.7
$\mathbf{A}_{os}^n$	369.7	593.9	820.0	1046.8
$p_n(\mathbf{A}_{ld})$	40.9	57.5	74.0	90.5
$p_n(\mathbf{A}_{bc})$	51.2	82.5	117.1	152.9
$p_n(\mathbf{A}_{os})$	55.4	93.7	134.8	176.5

C. Dependence on the parameters λ_M and λ_m

We have evaluated above the convergence speed of the new algorithm only considering the best parameters that include all the eigenvalues of the matrix. However, in most situations the nodes will have no knowledge about these eigenvalues. In this subsection we analyze the convergence rates of the algorithm when it is run using different parameters. We have only evaluated the algorithm with $\mathbf{A}_{ld}$.

The results of the experiment are in Table II. The experiment has considered again 100 different networks and for each one 100 different initial conditions. The table shows the average number of iterations required to have an error lower



(a) (b) (c)

Fig. 2. Numerical errors in the evaluation of the minimal polynomial lead to errors in the final estimation. (a) Communication network of 35 nodes. (b) Estimations of the 35 nodes of the network evaluating the minimal polynomial in a distributed way. The red dashed line shows the average of the initial conditions and the bars are the values of the final estimations. The rounding errors make the algorithm fail. (c) Evolution of the distributed evaluation of $p_n(\mathbf{A})\mathbf{x}(0)$. In this case the algorithm is stable and all the nodes' values converge to the real average (black dashed line). Moreover, for a tolerance error of 10^{-3} the algorithm requires less than N iterations to converge.

than 10^{-3}. The number of iterations is in all the cases larger than in Table I (57.5) but anyway, the results are in most cases still far better than those using (2). The only problem appears when $\lambda_M + \lambda_m - 1 > \lambda_N$ because the algorithm diverges. We have set a maximum of 1500 iterations to avoid the locks that appear in such situations.

Another advantage of using $\mathbf{A}_{ld}$ is that usually its smallest eigenvalue, λ_N, is a negative value close to zero (in our simulations it has never valued less than -0.5). The second largest eigenvalue depends on how many nodes has the network and the number of links, but in general this eigenvalue is close to one. Therefore by choosing $\lambda_m = -0.5$ and $\lambda_M \simeq 1$ there is a great chance to obtain a good convergence rate and almost no risk of divergence, see for example the cell in the second row and sixth column of Table II.

TABLE II
NUMBER OF ITERATIONS USING SUB-OPTIMAL PARAMETERS

$\lambda_m \backslash \lambda_M$	0.2	0.5	0.8	0.9	0.95	0.999
-0.2	582.2	474.7	334.3	208.6	$\geq$1500	$\geq$1500
-0.5	638.0	520.1	366.3	228.5	157.7	153.7
-0.8	689.1	561.9	395.7	246.9	170.3	165.4
-0.9	718.1	585.5	412.3	257.2	177.5	172.6
-0.95	727.5	593.1	417.7	260.6	179.8	174.9
-0.999	736.6	600.5	423.0	263.8	182.1	177.5

V. Conclusions

We have presented a new distributed consensus algorithm using Chebyshev polynomials. The proposed algorithm significantly reduces the number of communication rounds required by the network to achieve the consensus. We have provided a theoretical analysis of the convergence speed and the properties of the algorithm. We have also evaluated our method with an extensive set of simulations. Both theoretical and empirical analysis show the goodness of our proposal. In addition, we have shown that the evaluation of the minimal polynomial in large networks can be badly affected by numerical errors. Further research trying to extend our results to time varying networks and the distributed selection of the parameters in the algorithm is in progress.

Acknowledgments

The authors would like the reviewers for their insightful comments to improve the quality of the paper.

References

[1] F. Bullo, J. Cortés, and S. Martínez. *Distributed Control of Robotic Networks*. Applied Mathematics Series. Princeton University Press, 2009. Electronically available at http://coordinationbook.info.

[2] A. Jadbabaie, J. Lin, and A.S. Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Transactions on Automatic Control*, 48(6):988–1001, 2003.

[3] M. Zhu and S. Martínez. Discrete-time dynamic average consensus. *Automatica*, 46(2):322–329, 2010.

[4] M. Mehyar, D. Spanos, J. Pongsajapan, S. H. Low, and R. M. Murray. Asynchronous distributed averaging on communication networks. *IEEE/ACM Transactions on Networking*, 15(3):512–520, 2007.

[5] L. Xiao, S. Boyd, and S. Lall. A scheme for robust distributed sensor fusion based on average consensus. In *International Conference on Information Processing in Sensor Networks*, pages 63–70, 2005.

[6] J.C. Mason and D. C. Handscomb. *Chebyshev Polynomials*. Chapman and Hall, 2002.

[7] L. Xiao and S. Boyd. Fast linear iterations for distributed averaging. *Systems and Control Letters*, 53:65–78, 2004.

[8] Z. Jin and R.M. Murray. Multi-hop relay protocols for fast consensus seeking. In *IEEE Int. Conference on Decision and Control*, pages 1001 – 1006, 2006.

[9] B. Oreshkin, M. Coates, and M. Rabbat. Optimization and analysis of distributed averaging with short node memory. *IEEE Transactions on Signal Processing*, 58(5):2850–2865, May 2010.

[10] S. Muthukrishnan, B. Ghosh, and M. H. Schultz. First- and second-order diffusive methods for rapid, coarse, distributed load balancing. *Theory of Computing Systems*, 31(4):331–354, 1998.

[11] E. Kokiopoulou and P. Frossard. Polynomial filtering for fast convergence in distributed consensus. *IEEE Transactions on Signal Processing*, 57(1):342354, 2009.

[12] S. Sundaram and C. N. Hadjicostis. Finite-time distributed consensus in graphs with time-invariant topologies. In *American Control Conference*, pages 711–716, New York, 2007.

[13] Y. Yuan, G. Stan, L. Shi, and J. Gonçalves. Decentralised final value theorem for discrete-time lti systems with application to minimal-time distributed consensus. In *IEEE Int. Conference on Decision and Control*, pages 2664–2669, 2009.

[14] F. Jiang and L. Wang. Finite-time information consensus for multi-agent systems with fixed and switching topologies. *Physica D*, 238:1550–1560, 2009.

[15] J. Cortés. Finite-time convergent gradient flows with applications to network consensus. *Automatica*, 42(11):1993–2000, 2006.

[16] C.K. Ko and X. Gao. On matrix factorization and finite-time average-consensus. In *IEEE Int. Conference on Decision and Control*, pages 5798–5803, 2009.

[17] N. J. Higham. *Accuracy and Stability of Numerical Algorithms*. SIAM, 2002.